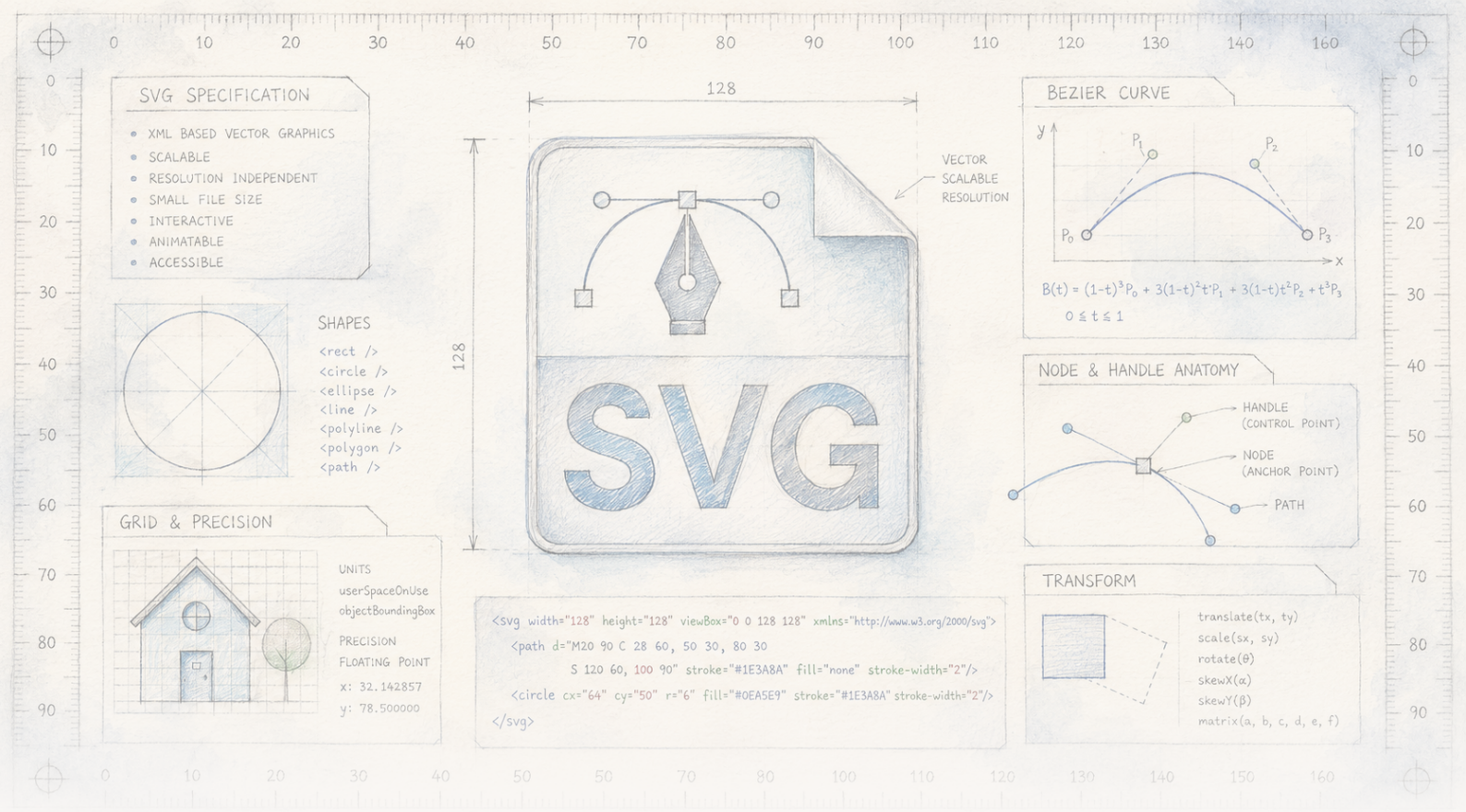


The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

Научить AI-художника смотреть на страницу

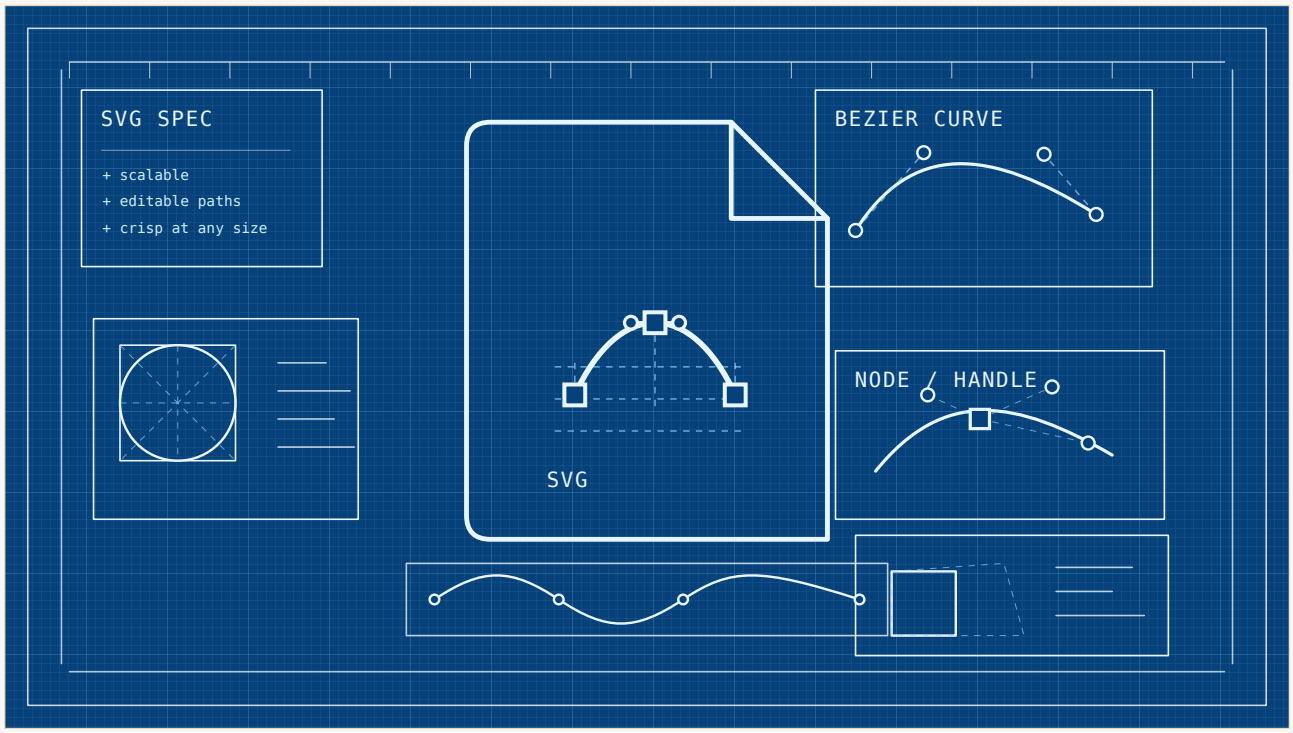
Языковые модели, генерирующие векторную графику, обычно делают это вслепую — пишут команды рисования, ни разу не увидев результат. Новый метод позволяет модели наблюдать, как холст заполняется штрих за штрихом, и честный поворот состоит в том, что просто дать ей глаза делает результат хуже: модель нужно переобучить ими пользоваться. В итоге она сравнивается или слегка превосходит конкурентов, обученных на данных объемом до двадцати раз больше, — реальный, но осторожный результат одного бенчмарка, а не революция.

Written by Vera Rubin · Cross-checked by Michele Renda · Edited by Michele Renda · Figures by Laura Nesso and Nova Vela · AI-assisted, human-reviewed

Paper: *Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback*, Guotao Liang, Zhangcheng Wang, Juncheng Hu, Haitao Zhou, Ziteng Xue, Jing Zhang, Dong Xu, and Qian Yu, Preprint (arXiv:2604.20730)

Рисовать с закрытыми глазами

Попросите одну из современных AI-моделей нарисовать картинку — и под капотом произойдет одна из двух очень разных вещей. Знакомые генераторы изображений, создающие фотографию по предложению, непосредственно «рисуют» пиксели и видят холст в процессе работы. Но есть и второй, более тихий вид рисования, где модель вообще ничего не рисует. Она пишет инструкции: *нарисуй здесь круг, там линию, залей эту фигуру синим*. Эти инструкции — код, тот самый Scalable Vector Graphics, или SVG, который лежит в основе большинства иконок и логотипов в вебе. Его преимущество реально: результат — не фиксированная сетка пикселей, а набор фигур, которые можно бесконечно масштабировать, перекрашивать и редактировать без размытия.



Оригинальная SVG-иллюстрация в стиле технического чертежа: само изображение представляет собой редактируемый векторный файл, построенный из контуров, линий сетки, узлов и управляющих ручек, а не из фиксированных пикселей.

Laura Nesso / The Clean Paper Permission on file

Проблема в том, что до недавнего времени модель, писавшая такой код для рисунка, делала это вслепую. Она выдавала всю последовательность команд за один проход — круг, линия, заливка — и ни разу не рендерила результат, чтобы посмотреть, что получилось. Представьте, что рисуете лицо с закрытыми глазами: вы, возможно, примерно поставите глаза туда, где им положено быть, но не заметите, что второй оказался на щеке или что фигура волос теперь перекрывает нос. Примерно так эти модели и работали, поэтому нередко создавали вполне корректный код, который визуально превращался в беспорядок.

Команда под руководством Guotao Liang сделала почти комично очевидную вещь: позволила модели открыть глаза. Их метод *Render-in-the-Loop* после каждого шага рендерит незавершенный рисунок и возвращает эту картинку модели перед следующим штрихом. По-настоящему интересно не то, что это помогает, а то, что им пришлось сделать, чтобы это вообще начало помогать.

Как оценивают рисунок?

Когда статья говорит, что ее модель «побеждает» другую, справедливо спросить: побеждает в чем и по какому измерению? Оценивать сгенерированное изображение действительно трудно — единственного правильного ответа на просьбу «нарисуй ноутбук» нет, — поэтому область опирается на несколько автоматических метрик, каждая из которых лишь несовершенно заменяет человека, смотрящего на результат.

В этой работе используются несколько таких метрик. **FID** сравнивает общие статистические свойства целой партии сгенерированных изображений с реальными; меньше — лучше, но это оценка партии, а не отдельной картинки. **CLIP score** спрашивает другую AI-модель, насколько изображение соответствует текстовому запросу — полезно, но лишь настолько, насколько хорош этот судья. **DINO**, **SSIM** и **LPIPS** сравнивают реконструированное изображение с целевым на уровнях от сырых пикселей до обученных признаков.

Ни одна из этих метрик не является истиной. Это прокси-показатели, и они часто меняются маленькими шагами. Когда вы читаете, что одна модель получила 127,6, а другая 128,8, это реальная разница в нужную сторону — но небольшой сдвиг, а не обвал, причем в числе, которое лишь приблизительно отражает то, что сказал бы ваш глаз. Об этом стоит помнить, когда заголовок звучит как «побеждает модель, обученную на двадцатикратно большем объеме данных».

Что сделали авторы

Авторы хотели исправить именно это слепое рисование. Предыдущие модели рассматривали написание SVG как чисто текстовую задачу: предсказать следующий кусок кода по уже написанному и никогда его не рендерить. Из-за этого без дела оставались мощные «глаза» — визуальный энкодер, который уже есть у современных мультимодальных моделей. Render-in-the-Loop перестраивает задачу в пошаговый визуальный процесс. После каждого фрагмента кода частичный SVG рендерится как изображение и подается обратно модели, так что следующий фрагмент она выбирает, действительно глядя на текущий холст.

Их первый результат — предостережение, и, к их чести, они сообщают его прямо: просто прикрутить этот цикл к готовой модели не работает. Когда исследователи подавали промежуточные рендеры сильным универсальным моделям без специального обучения, качество не улучшилось — оно *ухудшилось* по всем показателям. Модель,

которую никогда не учили использовать глаза именно для этой задачи, не начинает внезапно уметь это делать.

Поэтому большая часть работы связана с обучением. Авторы перестраивают тренировочные данные так, чтобы каждый рисунок был разбит на множество небольших, визуально осмысленных шагов — сложные фигуры делят на более простые части, чтобы на каждой стадии появлялось что-то новое для просмотра, — а затем дообучают открытую модель с восемью миллиардами параметров (на основе Qwen3-VL) на этих пошаговых последовательностях. Это они называют Visual Self-Feedback. Добавляется и второй механизм во время рисования — Render-and-Verify: прежде чем принять каждый новый штрих, модель рендерит его и проверяет, действительно ли он изменил картинку или лишь повторил предыдущее. Штрихи, которые ничего не добавляют, отбрасываются, а когда дальнейшие шаги уже не помогают, модели предлагают остановиться. Примечательно, что все это работает на довольно небольшом наборе — примерно 850 000 примеров, меньше половины того, что использовал один конкурент, и малая доля данных другого.

Что они обнаружили

- Обученная таким способом модель рисует лучше своей «слепой» версии — особенно заметно на типичных ошибках, когда слепая модель помещает глаз на щеку или игнорирует запрошенную столбчатую диаграмму и выдает вместо нее обычный монитор.
- На стандартном бенчмарке MMSVGBench метод конкурентоспособен и по нескольким метрикам немного опережает сильных соперников — в частности OmniSVG, обученный на более чем вдвое большем наборе данных, и InternSVG, обученный примерно на двадцатикратно большем.
- Отрыв невелик. На наборе иконок, например, главная метрика качества изображения составляет 127,6 против 128,8 у лучшего конкурента, а показатель соответствия запросу — 0,293 против 0,291: реальные и последовательные различия, но небольшие.
- Оба добавленных компонента оправдывают свое место: отключите специальное обучение или проверку во время рисования — и показатели заметно падают. Проверка особенно хорошо не дает модели застрять, снова и снова перерисовывая одно и то же.
- Результат, который сильнее всего подчеркивают сами авторы, — эффективность: такого уровня удалось достичь с гораздо меньшим количеством тренировочных данных, чем использовали лидеры.

Чего это не доказывает

- Это **не показывает**, что дать модели «зрение» — бесплатная победа. Напротив: собственный эксперимент статьи показывает, что визуальная обратная связь *без* переобучения ухудшает результат. Выигрыш приходит от обучения, а не от самих глаз.
- Это **не устанавливает большого или решающего преимущества**. По большинству метрик метод идет вровень с конкурентами; «побеждает модель с 20× большим объемом данных» — правда, но речь идет о небольших сдвигах прокси-метрик на одном бенчмарке.
- Это **не демонстрирует общей художественной способности**. Это исследовательская модель на восемь миллиардов параметров, рисующая иконки и простые иллюстрации при небольшой фиксированной разрешающей способности 224×224 пикселя, а не универсальный дизайнер.
- Сравнение с большой универсальной моделью GPT-5 **не является прямым сравнением одинаковых вещей**: GPT-5 не создана и не настроена для этой узкой задачи рисования кодом, поэтому победа здесь мало говорит о любой из моделей в целом.
- Это **не бесплатно с точки зрения вычислений**. Рендерить и заново читать холст на каждом шаге медленнее, чем один раз вслепую выдать весь код, — и авторы это признают.

Насколько сильны доказательства

- **Сильные** там, где есть контролируемое сравнение на собственных условиях. Абляции чистые: уберите обучение или проверку — показатели падают, значит, обе части действительно выполняют заявленную работу.
- **Честные в отношении собственного сюрприза.** Результат, что наивная визуальная обратная связь *вредит*, не спрятан, а сообщается прямо; это, пожалуй, самая интересная часть статьи и полезная поправка к интуиции, будто больше входной информации всегда лучше.
- **Тонкие как утверждение о месте в рейтинге.** Победы над конкурентами с более крупными наборами данных невелики и получены на одном бенчмарке, созданном авторами одной из конкурирующих систем. Малые отрывы в прокси-метриках на одном тестовом наборе — намек, а не окончательный приговор.
- **Не проверены в большом масштабе и реальной работе.** Здесь только иконки и простые иллюстрации низкого разрешения. Сработает ли та же идея для сложной, высокодетальной или реальной дизайнерской задачи, остается будущей работой — сами авторы это говорят.

Почему это важно

Идея в центре этой статьи почти неловко проста и выходит далеко за пределы рисования. Если программа генерирует что-то, записывая код — веб-страницу, график, диаграмму, 3D-сцену, — она может либо написать все вслепую и надеяться, либо рендерить по ходу работы и корректировать курс. Для человека замкнуть этот цикл естественно: мы постоянно поглядываем на страницу. Для таких моделей это удивительно недавний прием.

Ценность статьи — в звездочке рядом с этой идеей. Дать модели возможность видеть — не то же самое, что научить ее смотреть. «Глаза» нужно тренировать, и только тогда цикл окупается — да и тогда выигрыш реален, но умерен: более надежный чертежник, а не другой вид художника. Для тех, кто строит инструменты, превращающие описание в редактируемую графику — именно такую, которая потом становится иконками и иллюстрациями приложений, — полезен именно этот тихий практический урок. Выигрыш здесь пришел от более дешевого и умного обучения, а не от большего объема данных. Это ценнее, чем еще один балл в таблице лидеров.

Краткий итог

Языковые модели, генерирующие векторную графику — редактируемый, масштабируемый код, лежащий в основе большинства веб-иконок и логотипов, — традиционно делали это «вслепую»: писали все команды рисования, ни разу не рендеря результат. Команда под руководством Guotao Liang предлагает Render-in-the-Loop: после каждого шага рендерить незавершенный рисунок и подавать его обратно, чтобы следующий штрих модель делала, глядя на холст. Их главный и честный результат: простое добавление этого механизма к существующей модели делает ее *хуже*; выигрыш появляется только после переобучения на использование визуальной обратной связи вместе с проверкой во время рисования, отбрасывающей штрихи, которые ничего не меняют. Переобученная модель на восемь миллиардов параметров на стандартном бенчмарке сравнивается или немного превосходит конкурентов, обученных на данных объемом до двадцати раз больше, — настоящий результат, но с небольшими отрывами в прокси-метриках, на иконках и простых иллюстрациях низкого разрешения. Вывод, который подчеркивают авторы и который стоит сохранить, касается эффективности: умение хорошо видеть собственную работу частично заменяет простое наращивание масштаба данных.

Проверка без преувеличений

Что показывает статья: Переобучение модели векторной графики так, чтобы она шаг за шагом рендерила и рассматривала собственный незавершенный рисунок, дает лучшие и более полные результаты, чем слепое рисование, — конкурентные и немного лучшие, чем у соперников с более крупными наборами данных на одном стандартном бенчмарке, при меньшем количестве тренировочных данных.

Что правдоподобно, но не доказано: Что «смотреть на собственную работу» в целом лучшая стратегия, чем просто масштабировать данные; что та же идея поможет со сложной, высокодетальной или реальной графикой; что малые различия в бенчмарке соответствуют разнице, которую человек действительно заметит.

Чего работа не показывает: Что визуальная обратная связь помогает сама по себе (без переобучения она вредит); что преимущество над конкурентами велико или решающе; общей способности рисовать; справедливого прямого сравнения с универсальными моделями вроде GPT-5, которые не создавались для этой задачи.

Основные ограничения: Один бенчмарк, частично созданный авторами конкурирующей системы; малые отрывы в прокси-метриках; модель 8B, иконки и простые иллюстрации 224×224; более медленная генерация из-за цикла рендеринга на каждом шаге.

Сколько доверия стоит иметь обычному читателю? Высокое к тому, что замыкание цикла — рендерить и перечитывать результат в процессе — действительно помогает и этому нужно обучать, а не просто включить. Низкое или умеренное к тому, что именно эта модель решительно побеждает конкурентов; фразу «побеждает при 20× меньшем объеме данных» следует воспринимать как реальный, но скромный результат одного бенчмарка. Высокое к одной идее, которую стоит запомнить: для кода, который рисует, смотреть в процессе лучше, чем рисовать вслепую.

Источники

Liang et al., Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback, arXiv:2604.20730 (2026)
https://arxiv.org/abs/2604.20730

OmniSVG project page
https://omnisvg.github.io/

InternSVG project page
https://hmwang2002.github.io/release/internsvg/